Robust and efficient solvers for large and indefinite linear systems

Ichitaro Yamazaki   Xiaoye Li   Esmond Ng

Lawrence Berkeley National Laboratory

ComPASS collaboration meeting,
UCLA, December 3, 2008

In many ComPASS EM applications, solving linear systems is the "memory" bottleneck;

$$Ax = b,$$

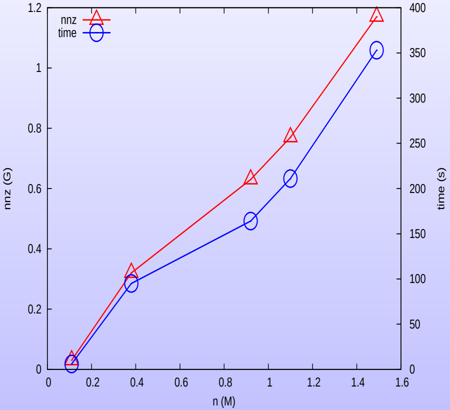
where A is

- large and sparse
  - direct methods are robust, but require infeasibly large memory.
- ill-conditioned and highly-indefinite
  - preconditioned iterative methods require less memory, but suffer from slow or no convergence.

Hybrid methods have the potential of balancing memory and time;

- techniques from direct methods are used to transform the original system into a "smaller" system, which is "easier" to solve by iterative methods.

Direct method: SuperLU_DIST with MeTiS

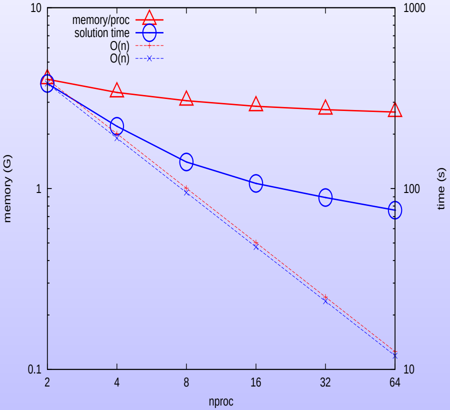


	n (M)	nnz (G)	fill-ratio	time (s)
dds	0.02	0.02	4	6
tdr108	0.11	0.03	18	6
quad	0.38	0.32	21	95
tdr158	0.92	0.63	17	164
tdr190	1.10	0.77	18	211
tdr256	1.49	1.17	20	353

two processes running on two nodes of Franklin.

- large amount of fill: **tdr256k** could not be solved on one node (which has 7.38GB of memory).

Direct method: SuperLU_DIST with MeTiS

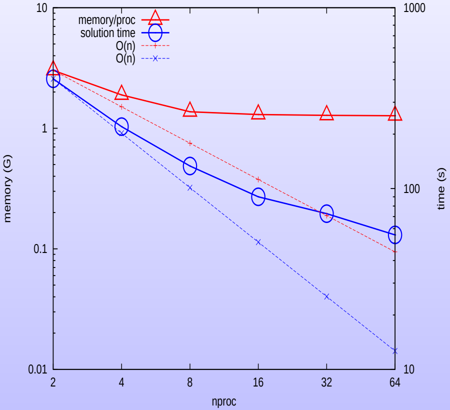


nproc	mem (GB)	time (s)	speed up
2	4.02	380.95	1.00
4	3.40	221.06	1.72
8	3.06	140.26	2.72
16	2.85	102.81	3.71
32	2.73	89.18	4.27
64	2.65	75.97	5.01

tdr256k performance profiled using Craypat on Franklin.

- memory requirement scales poorly with the number of processors.
  - each process required an explicit use of entire memory on a node.

Direct method: SuperLU_DIST with ParMeTiS



nproc	mem (GB)	time (s)	speed up
2	3.02	404.42	1.00
4	1.89	219.98	1.83
8	1.37	133.26	3.03
16	1.30	90.00	4.49
32	1.28	72.65	5.57
64	1.27	55.43	7.30

tdr256k performance profiled using Craypat on Franklin.

- parallel permutation and symbolic factorization improve scalability.
- memory requirement can still be the bottleneck.

Preconditioned iterative method: Phidal [Henon and Saad '06]

- GMRES(100) to achieve $\|b - Ax\|_2 / \|b\|_2 \leq 10^{-12}$ on one core of Franklin.

drop tol.	fill-ratio	itrs	ptime	stime	ttime
SuperLU	17.52	$--$	7.54	0.39	7.93
10^{-5}	14.85	17	10.79	2.64	13.47
10^{-4}	12.84	> 1000	7.60	$--$	$--$

$105,386 \times 105,386$ **tdr108k**, times are in seconds.

drop tol.	fill-ratio	itrs	ptime	stime	ttime
SuperLU	20.53	$--$	167.57	2.03	169.60
10^{-4}	19.95	50	732.70	77.07	809.77
10^{-3}	19.91	> 1000	743.04	$--$	$--$

$380,698 \times 380,698$ **dds-quad**, times are in seconds.

- no convergence with a relatively large number of nonzeros.
- larger system often requires a larger fill-ratio for convergence.

Hybrid method: Schur complement method

Step 1: Reorder A into a 2×2 block system of the form

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

where

- A_{11} is $n_1 \times n_1$ block-diagonal, and A_{22} is $n_2 \times n_2$.
- A_{11} is typically referred to as the *interior domains*, A_{22} is called the *separators*, and A_{21} and A_{12} are the *interfaces* between A_{11} and A_{22}.
- Existing software like ParMeTiS, PT-SCOTCH, or HID can be used to create the block structure.

Hybrid method: Schur complement method

Step 1: With a block Gaussian elimination, the 2×2 block system becomes

$$\begin{pmatrix} A_{11} & A_{12} \\ 0 & S \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} I & 0 \\ -A_{21}A_{11}^{-1} & I \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \end{pmatrix},$$

where $S = A_{22} - A_{21}A_{11}^{-1}A_{12}$ is called the Schur complement.

Hence, the solution to the linear system is given by

- $Sx_2 = b_2 - A_{21}A_{11}^{-1}b_1$ (**Steps 2** and **3**), and
- $A_{11}x_1 = b_1 - A_{12}x_2$ (**Step 4**).

Hybrid method: Schur complement method
Step 2: Solve the system of interior domains;

$$A_{11} z_1 = b_1$$

for z_1, based on an exact LU factorization of A_{11},

$$A_{11} \rightarrow L_1 U_1,$$

where L_1 is lower-triangular and U_1 is upper-triangular.

- Each diagonal block can be factored and solved independently using software like SuperLU_DIST.
- Appropriate scaling and permutation are applied to enhance numerical stability and preserve sparsity.

Hybrid method: Schur complement method
Step 3: Approximately solve

$$Sx_2 = \widehat{b}_2$$

for x_2, where S is the **Schur complement**, and $\widehat{b}_2 = b_2 - A_{21}z_1$.

Krylov method (e.g. GMRES) is used with a preconditioner based on an incomplete factorization of S:

$$\begin{cases} S &= \widetilde{S} + E_S, \\ \widetilde{S} &= \widetilde{L}_2\widetilde{U}_2 + E_{LU}, \end{cases}$$

where

- E_S and E_{LU} are error matrices.
- Sparsity of $\widetilde{S}$, $\widetilde{L}_2$, and $\widetilde{U}_2$ is enforced by discarding small nonzeros.
- S is accessed only through matrix-vector multiply.

Hybrid method: Schur complement method

Step 4: Solve the system of interior domains;

$$A_{11}x_1 = \widehat{b}_1,$$

where $\widehat{b}_1 = b_1 - A_{12}x_2$, and the factorization of A_{11} from **Step 2** is used.

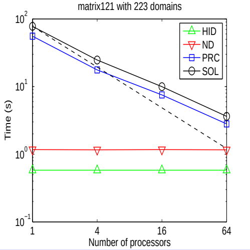
Our **motivations** for focusing on this hybrid method is

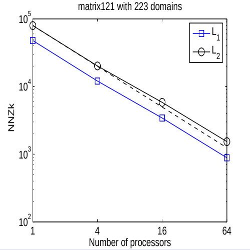
- memory requirement: fill is restricted within
  - "small" diagonal blocks of L_1 and U_1, and
  - $\widetilde{S}$, $\widetilde{L}_2$, and $\widetilde{U}_2$, whose "sparsity" can be enforced.
- conditioning: in comparison to A,
  - S is smaller in its dimension, and
  - S often has more favorable eigenvalue distribution.

  The linear system with S is expected to be "easier" to solve using a preconditioned iterative method
- parallelism: interior domains are solved independently.

HIPS (Hybrid Iterative Parallel Solver):

- Developed by Pascal Henon (INRIA) and Yousef Saad (UM), 2008.
- Based on HID to achieve scalability on a parallel machine.





- Current limitations:
  - Number of processors cannot exceed the number of interior domains.
  - Fill in ILU($\widetilde{S}$) is restricted within the "level-1" blocks of A_{22}.

  To run on many processors, many interior domains are needed, which results in slow or no convergence
  - large Schur complement and poor preconditioner (fill is restricted within small blocks).

Results of HIPS

tol	dom	n_2	nnz	fill-fact	itrs	ptime	stime	ttime
direct	1	$--$	13M	16.8	$--$	3.22	0.14	3.41
10^{-4}	2	54	13M	16.7	4	3.77	0.64	4.47
	4	238	13M	16.3	6	3.78	0.80	4.64
	8	1774	10M	13.5	24	3.35	2.14	5.57
	15	3056	9M	12.1	56	3.72	4.65	8.44
	32	6091	8M	10.1	680	4.76	58.44	63.31
	61	9556	7M	8.8	> 2000	5.35	$--$	$--$
10^{-6}	32	6091	8M	11.0	85	9.12	7.85	17.08
	61	9556	7M	9.8	> 2000	8.66	$--$	$--$

$105,386 \times 105,386$ **tdr108k** with $\mathrm{nnz}(A) = 804,301$.

- Large number of domains is needed to reduce memory cost.
- GMRES(100) does not converge with a large number of domains.
- Larger system requires a smaller number of domains for convergence, i.e., for **tdr256k**, HIPS did not converge with 4 interior domains.

New implementation of the hybrid method, whose performance is better than HIPS. Our goals are to

- improve numerical stability,
- solve each interior domain with multiple processors, and
- implement our own parallel ILU.

Implementation approach:
we rely on existing software (performance, development time, etc.);

- **Step 1**: initial partitioning is computed by HID.
- **Step 2**: system of interior domains is solved using SuperLU_DIST.
- **Step 3**: drop tolerance σ_1 enforces sparsity of $\widetilde{S}$, and drop tolerance σ_2 enforces sparsity of $\widetilde{L}_2$ and $\widetilde{U}_2$.

	σ_1	σ_2	ILU/LU	Solve
(1)	zero	zero	SLU	SLU
(2)	nonzero	zero	SLU	PETSc
(3)	zero	nonzero	PHIDAL	HIPS
(4)	nonzero	nonzero	PHIDAL	PETSc

Four configurations to solve the Schur complement.

- **Step 4**: system of interior domains is solved by SuperLU_DIST, using their LU factorizations from Step 2.

Preliminary results of new implementation

	σ_1	σ_2	nnz	fill-ratio	itrs	ptime	stime	ttime
(1)	0.0	0.0	5.5M	4.0	−−	3.02	0.05	3.07
(4)	10^{-4}	10^{-4}	0.6M	0.5	300	1.50	8.47	9.97

$17,732 \times 17,732$ **dds** with 147 domains, times are in seconds.

	σ_1	σ_2	nnz	fill-ratio	itrs	ptime	stime	ttime
(1)	0.0	0.0	24M	16.8	−−	18.53	0.51	19.04
(4)	10^{-2}	10^{-4}	22M	15.6	64	71.92	43.78	115.70

$105,386 \times 105,386$ **tdr108k** with 468 domains, times are in seconds.

- GMRES(100) converged with the new implementation.
  - HIPS did not converge for **dds** and **tdr108k** with 25 and 61 domains, respectively.
- For a large system, Phidal and SuperILU require large amount of fill.

Preliminary results of new implementation

		$\mathrm{LU}(A_{11})$		$\mathrm{ILU}(\widetilde{S})$	
n_1	n_2	nnz	ratio	nnz	ratio
79,278	26,108	1.4M	3.2	20.6M	125.1

memory requirements for **tdr108k**.

$\mathrm{LU}(A_{11})$	$\mathrm{Comp}(\widetilde{S})$	$\mathrm{ILU}(\widetilde{S})$	Solve	Total
1.16	6.93	63.83	43.78	115.70

time requirements in seconds for **tdr108k**.

- $\mathrm{ILU}(\widetilde{S})$ is the memory bottleneck.
- time for $\mathrm{ILU}(\widetilde{S})$ and Solve is large due to large amount of fill.
- $\mathrm{Comp}(\widetilde{S})$ can be the bottleneck for a large matrix.

Current work:

- improving the convergence rate for solving the system of Schur complement
  - improving the quality of preconditioner, i.e., drop tolerance based ILU.
  - controlling the conditioning of the Schur complement.
- improving the time efficiency for computing Schur complement.
  - symbolic factorization to take advantage of sparisity.
- developing parallel implementation.
- conducting further experimentation.

Extra slides

Schur complement computation:

S is computed domain-by-domain,

$$
\begin{aligned}
& A_{22} - A_{21} A_{11}^{-1} A_{12} \\
&= A_{22} - \begin{pmatrix} A_{21}^{(1)} & A_{21}^{(2)} & \dots & A_{21}^{(\ell)} \end{pmatrix} \begin{pmatrix} A_{11}^{(1)} & & & \\ & A_{11}^{(2)} & & \\ & & \ddots & \\ & & & A_{11}^{(\ell)} \end{pmatrix}^{-1} \begin{pmatrix} A_{12}^{(1)} \\ A_{12}^{(2)} \\ \vdots \\ A_{12}^{(\ell)} \end{pmatrix} \\
&= A_{22} - \sum_{i=1}^{\ell} A_{21}^{(i)} B_i^{-1} A_{12}^{(i)}.
\end{aligned}
$$

- each $A_{11}^{(i)}$ is scaled and permuted for numerical stability.
- each $A_{21}^{(i)}$ and $A_{12}^{(i)}$ are stored in the CSR and CSC formats.
- S is formed by k columns at a time and stored in the CSC format.

For a symmetric A,

Schur complement is computed as

$$
\begin{aligned}
& A_{22} - \textstyle\sum_{i=1}^{\ell} A_{21}^{(i)} A_{(11)}^{(i)-1} A_{12}^{(i)} \\
&= A_{22} - \textstyle\sum_{i=1}^{\ell} (L_i^{-1} A_{12}^{(i)})^T (L_i^{-1} A_{12}^{(i)}) \\
&= A_{22} - \textstyle\sum_{i=1}^{\ell} W_i^T W_i
\end{aligned}
$$

- eliminates one triangular solve
  - computation time can be reduced to about half.
- requires memory to store W_i
  - only one W_i needs to be stored at a time.
- takes advantage of sparsity of W_i
  - sparsity of W_i can be enforced.

Preprocessing:
S is preprocessed to minimize possibility of a singular $\widetilde{S}$:

$$S = P_r D_r S D_c,$$

- P_r is a row permutation to move large elements to the diagonals.
- D_c and D_r are column and row scaling matrices, respectively such that S has unit diagonals.
- this preprocess is efficiently applied since S is stored in CSC.

The subroutine **mc64ad** developed by Iain Duff is used.

Compute ILU of $\widetilde{S}$:

- enforce sparsity of $\widetilde{S}$.
  - discard nonzeros of S with magnitudes less than a drop tolerance σ_1.
- compute LU or ILU factorization of $\widetilde{S}$ using SuperLU or Phidal.
  - SuperILU based on level computation after the symbolic factorization of each columns of $\widetilde{L}_2$ and $\widetilde{U}_2$.
- free memory used to store $\widetilde{S}$.

	nnz	fill-ratio	itrs	ptime	stime	ttime
Direct	287M	19.51	−−	579.53	1.66	581.19
Phidal(10^{-4})	277M	18.92	59	2,026.35	61.88	2,088.23
SuperILU(2)	239M	15.28	4	802.88	9.52	812.40

$380,698 \times 380,698$ **dds-quad** with 284 domains.

- for **dds-quad**, HIPS did not converge for 8 domains.